

8º CAPÍTULO

Segmentação de imagem

Prof. Arnaldo Abrantes / anotado por Prof. Nuno Pinho da Silva

- Objectivo: decompôr uma imagem (ou sequência de imagens) num conjunto de regiões (1d, 2d ou 3d), podendo obter-se
 - Áreas com um dado significado semântico, ou
 - Conjuntos de *pixels* de fronteira, ou
 - Estruturas de *pixels* com forma determinada (linhas, círculos, polígonos)
- Porquê realizar a operação de segmentação?
 - Isolar zonas de interesse, para posterior processamento.
 - Mudança de representação da imagem, para um nível hierárquico superior

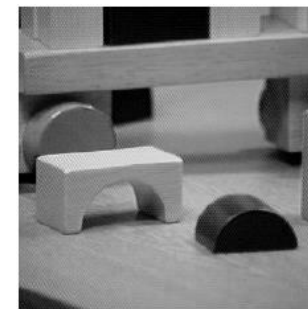
Organizar os píxeis em classes semânticas ou descritores mais apropriados para o processamento posterior.



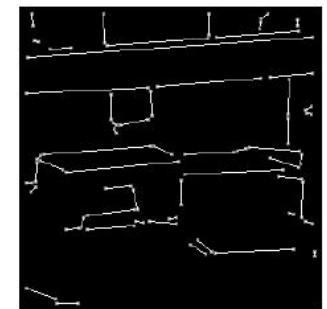
entrada



saída




entrada

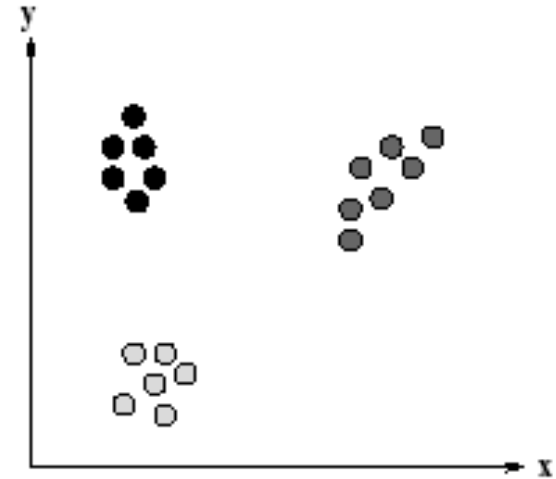


saída

O que é uma região?

- Conjuntos de *pixels* conexos com ‘propriedades’ homogêneas, relativamente à(s) característica(s) (*features*) usadas na operação de segmentação (nível de cinzento, cor, textura, etc.);
- Regiões adjacentes deverão ser significativamente diferentes, relativamente a essas mesmas *features*;
- As fronteiras das regiões deverão ser suaves; 
- O interior duma região deverá ser simples, não devendo conter um grande número de buracos;

- *Clustering*
 - Processo de decompôr (classificar) um conjunto de padrões (vetores de *features*) em diferentes subconjuntos, designados por *clusters*
- Exemplos de *features*
 - Intensidade
 - Cor
 - Textura
 - Forma
 - Movimento
- Alguns métodos
 - Algoritmos clássicos: K-médias, Isodata (Iterative Self-Organizing Data Analysis Techniques)
 - Baseados em histogramas: Otsu, Ohlander
 - Baseados em partições de grafos: Shi



Objectivo: $D = \sum_{k=1}^K \sum_{x_i \in C_k} \|x_i - m_k\|^2$ \longrightarrow

Form K-means clusters from a set of n-dimensional vectors.

1. Set ic (iteration count) to 1.
2. Choose randomly a set of K means $m_1(1), m_2(1), \dots, m_K(1)$.
3. For each vector x_i compute $D(x_i, m_k(ic))$ for each $k = 1, \dots, K$ and assign x_i to the cluster C_j with the nearest mean.
4. Increment ic by 1 and update the means to get a new set $m_1(ic), m_2(ic), \dots, m_K(ic)$.
5. Repeat steps 3 and 4 until $C_k(ic) = C_k(ic+1)$ for all k .



K = 6

Form isodata clusters from a set of n-dimensional vectors.

1. assign x_i to the cluster l that minimizes

$$D_{\Sigma} = [x_i - m_l]^T \Sigma_l^{-1} [x_i - m_l].$$

2. Merge clusters i and j if

$$|m_i - m_j| < \tau_v$$

where τ_v is a variance threshold.

3. Split cluster k if the maximum eigenvalue of σ_k is larger than τ_v .
4. Stop when

$$|m_i(t) - m_i(t+1)| < \epsilon$$

for every cluster i or when the maximum number of iterations has been reached.



K = 5

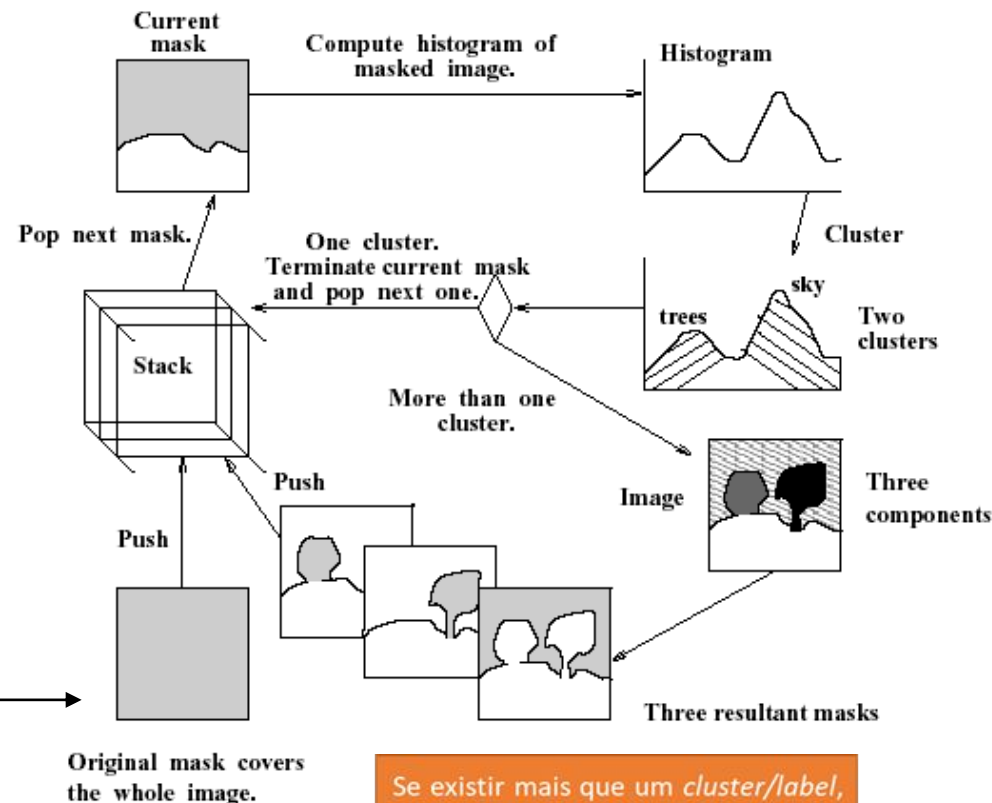
and Σ_k is defined by

$$\Sigma_k = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \dots & \sigma_{1n} \\ \sigma_{12} & \sigma_{22} & \dots & \sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{1n} & \sigma_{2n} & \dots & \sigma_{nn} \end{bmatrix} \quad (10.1)$$

where $\sigma_{ii} = \sigma_i^2$ is the variance of the i th component v_i of the vectors and $\sigma_{ij} = \rho_{ij}\sigma_i\sigma_j$ is the covariance between the i th and j th components of the vectors. (ρ_{ij} is the correlation coefficient between the i th and j th components, σ_i is the standard deviation of the i th component, and σ_j is the standard deviation of the j th component.)

Métodos baseados em histogramas

- Hipótese
 - Os subconjuntos de *pixels* que originam os diferentes modos do histograma, correspondem a diferentes objectos na cena (caso binário – método de Otsu)
- Vantagem
 - Métodos deste tipo necessitam de apenas uma passagem pelos dados
- Exemplo
 - Algoritmo recursivo de Ohlander



As máscaras no stack representam regiões candidatas a serem segmentadas

Se existir mais que um *cluster/label*, a extração de componentes conexos (ECC) é aplicada a cada cluster/label, resultando num conjunto de regiões (conexas, claro!) para cada cluster/label. Cada região gera uma máscara que vai para o stack.

Partição de grafos – algoritmo de Shi

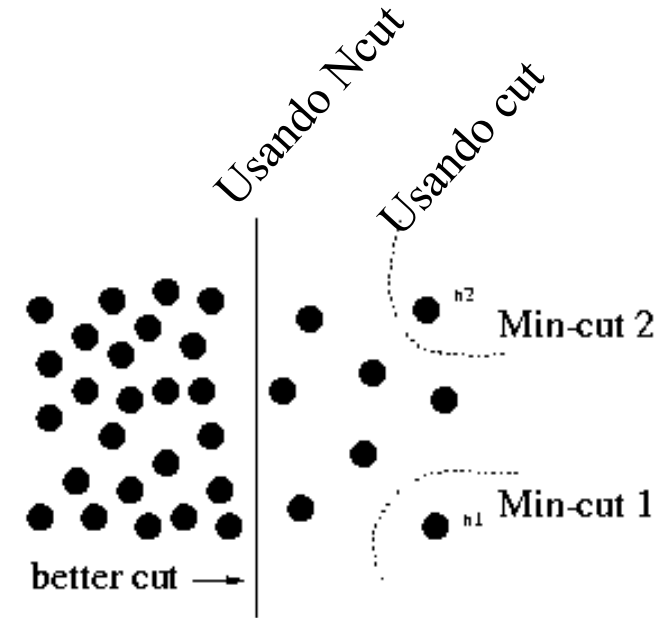
- Seja o grafo $G = (V, E)$ definido pelo conjunto de nós V e ligações E

Aplicável a qualquer espaço de features para o qual exista uma medida de semelhança

Cada nó representa um ponto no espaço de *features*

Cada ligação tem um peso, $w(i, j)$, que representa o grau de semelhança entre os nós i e j

- Problema da segmentação
 - Encontrar partições do grafo, tais que as medidas de semelhança intra-partições sejam elevadas, e as inter-partições baixas
- Caso binário



(pesos inversamente proporcionais à distância entre nós)

$$\text{cut}(A, B) = \sum_{u \in A, v \in B} w(u, v)$$

$$\text{asso}(A, V) = \sum_{u \in A, t \in V} w(u, t)$$

- Caso M-ário: recursão do algoritmo binário

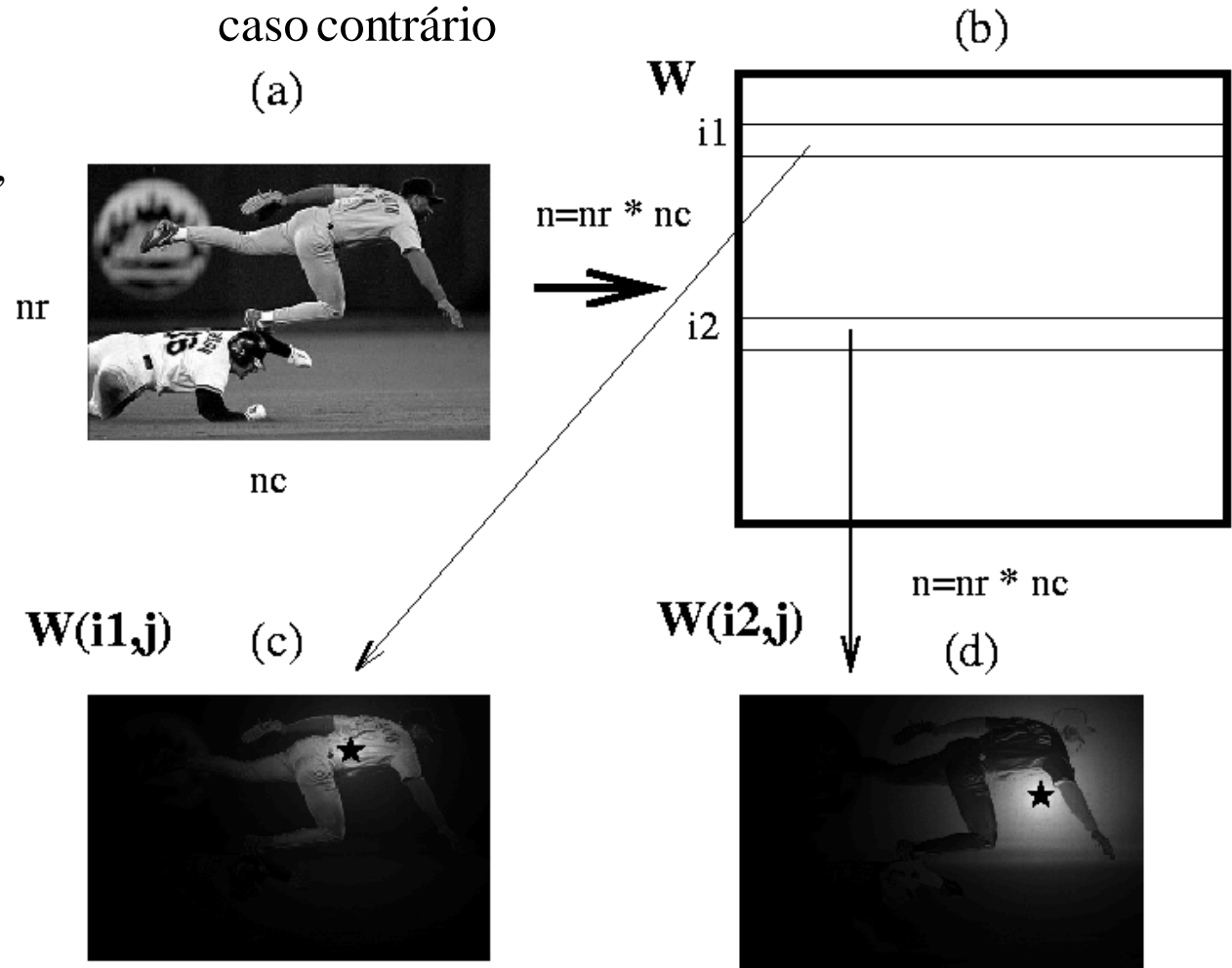
$$\text{Ncut}(A, B) = \frac{\text{cut}(A, B)}{\text{asso}(A, V)} + \frac{\text{cut}(A, B)}{\text{asso}(B, V)}$$

↑
Minimizar Ncut

Exemplo – Escolha da função de pesos

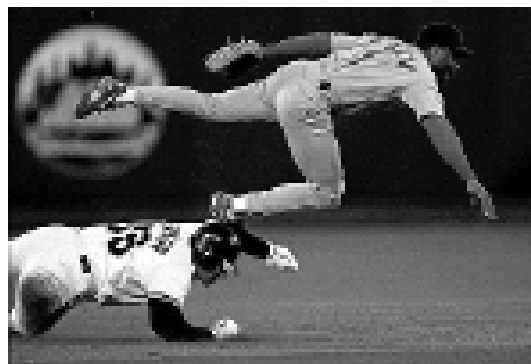
$$w(i, j) = e^{-\frac{\|F(i) - F(j)\|_2}{\sigma_I}} * \begin{cases} e^{-\frac{\|X(i) - X(j)\|_2}{\sigma_X}} & \text{se } \|X(i) - X(j)\|_2 < r \\ 0 & \text{caso contrário} \end{cases}$$

- $F(i), F(j)$ são vectores de *features*,
(por exemplo intensidade luminosa,
ou resposta a um banco de filtros)
- $X(i), X(j)$ são as correspondentes
coordenadas dos *pixels* i, j

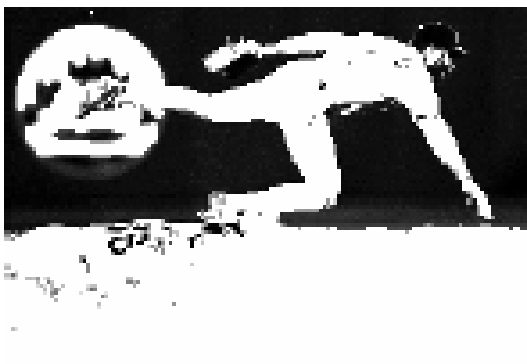


Resultados de segmentação – trabalho de Shi e Malik

original



fundo



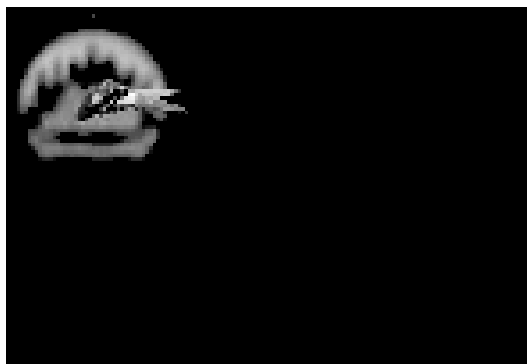
player1



player2



símbolo



solo



- ↑
- Problema de optimização formulado como um problema de cálculo de vectores e valores próprios

- Ideia
 - Começar com uma região de um só pixel (semente, *e.g.* canto superior esquerdo) e ir agrupando pixéis a essa região enquanto a medida de semelhança for suficientemente elevada;
 - Iniciar uma nova região quando falha;

66	71	66	52	38	22	16	19
65	61	48	33	29	25	24	24
49	36	20	13	15	16	14	11
56	47	40	40	40	36	31	29
108	105	104	105	105	103	97	95
129	131	134	136	137	136	137	140
131	130	129	130	131	130	131	134
129	128	125	127	128	127	127	129



Critério – diferença para o valor de intensidade do pixel *seed* ≤ 25

66	71	66	52	38	22	16	19
65	61	48	33	29	25	24	24
49	36	20	13	15	16	14	11
56	47	40	40	40	36	31	29
108	105	104	105	105	103	97	95
129	131	134	136	137	136	137	140
131	130	129	130	131	130	131	134
129	128	125	127	128	127	127	129

- Critério

- Distância entre pixéis

$$\begin{cases} d(p_1, p_2) < \tau & \text{semelhante} \\ d(p_1, p_2) \geq \tau & \text{não semelhante} \end{cases}$$

- Critério estatístico

$$T = \left[\frac{(N-1)N}{N+1} (y - \bar{X})^2 / S^2 \right]^{\frac{1}{2}} \quad \begin{array}{ll} T < \tau & \text{semelhante} \\ T \geq \tau & \text{não semelhante} \end{array}$$

- Atualizar, recursivamente, a média e a medida de dispersão

Se for
semelhante:

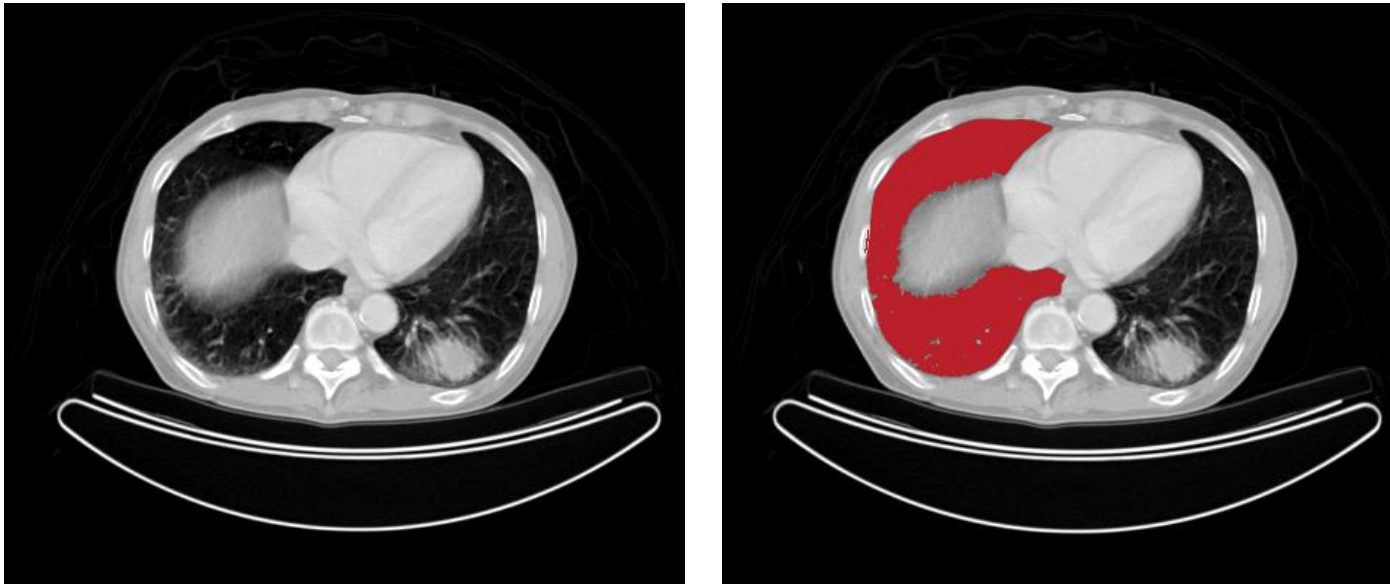
$$\bar{X}_{new} \leftarrow (N\bar{X}_{old} + y) / (N + 1)$$

$$\bar{X} = \frac{1}{N} \sum_{(r,c) \in R} I(r,c)$$

$$S_{new}^2 \leftarrow S_{old}^2 + (y - \bar{X}_{new})^2 + N(\bar{X}_{new} - \bar{X}_{old})^2$$

$$S^2 = \sum_{(r,c) \in R} (I(r,c) - \bar{X})^2$$

Segmentation by growing a region from seed point using intensity mean measure



<http://www.mathworks.com/matlabcentral/fileexchange/19084-region-growing>

- Algoritmo com 2 passos
 - *Top-down*
 - Partir sucessivamente a imagem em quadrantes enquanto um critério de homogeneidade não for cumprido (*quadtrees*);
 - *Bottom-up*
 - Juntar regiões adjacentes que satisfaçam um critério de semelhança;

Partição e Junção de Regiões

Critério de homogeneidade – Partição: variância de 1;
Junção: valor igual;

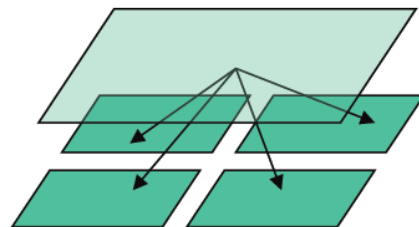
0	1	0	0	7	7	7	7
1	0	2	2	7	7	7	7
0	2	2	2	7	7	7	7
4	4	2	2	7	7	7	7
0	0	1	1	3	3	7	7
1	1	2	2	3	7	7	7
2	4	3	0	5	7	7	7
2	3	3	5	5	0	7	7

original image



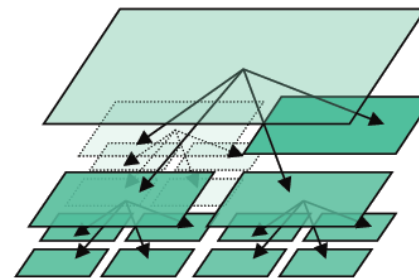
0	1	0	0	7	7	7	7
1	0	2	2	7	7	7	7
0	2	2	2	7	7	7	7
4	4	2	2	7	7	7	7
0	0	1	1	3	3	7	7
1	1	2	2	3	7	7	7
2	4	3	0	5	7	7	7
2	3	3	5	5	0	7	7

split 1



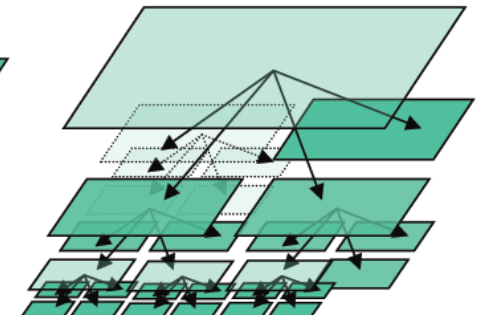
0	1	0	0	7	7	7	7
1	0	2	2	7	7	7	7
0	2	2	2	7	7	7	7
4	4	2	2	7	7	7	7
0	0	1	1	3	3	7	7
1	1	2	2	3	7	7	7
2	4	3	0	5	7	7	7
2	3	3	5	5	0	7	7

split 2



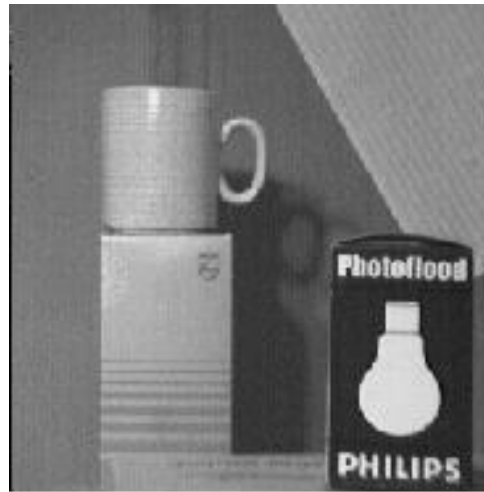
0	1	0	0	7	7	7	7
1	0	2	2	7	7	7	7
0	2	2	2	7	7	7	7
4	4	2	2	7	7	7	7
0	0	1	1	3	3	7	7
1	1	2	2	3	7	7	7
2	4	3	0	5	7	7	7
2	3	3	5	5	0	7	7

split 3



Adaptado de <http://uei.ensta.fr/baillie>

Partição e Junção de Regiões



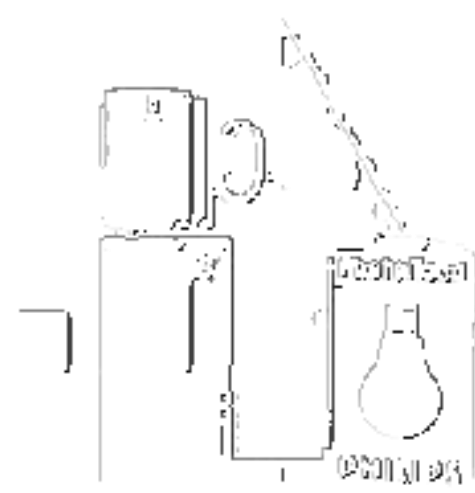
Original image



After quad splitting



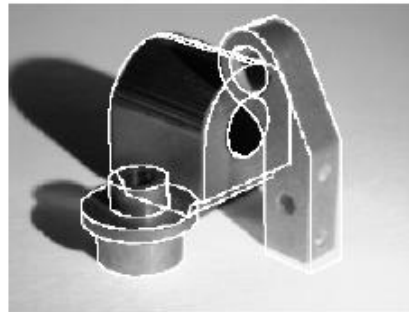
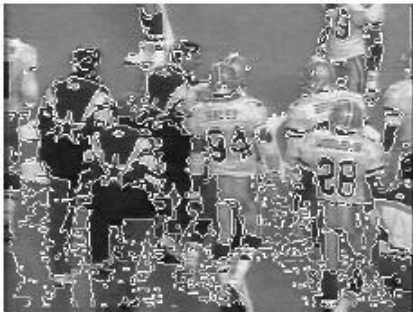
After merging



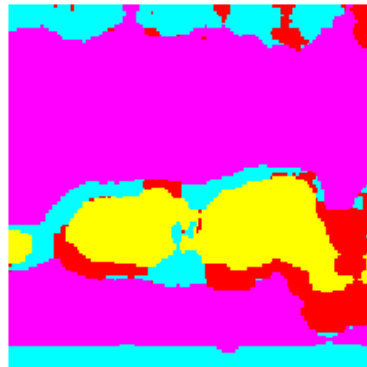
Edge delineation of regions

Representação de regiões

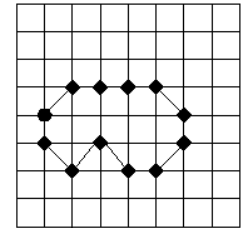
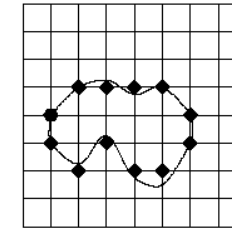
- *Overlays*



- Imagem de *labels*

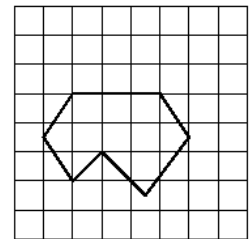
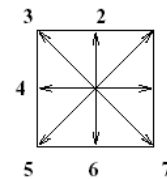


- *Chain codes*



original curve

chain code links



100076543532

chain code representation

polygonal approximation

- *Quadtrees*

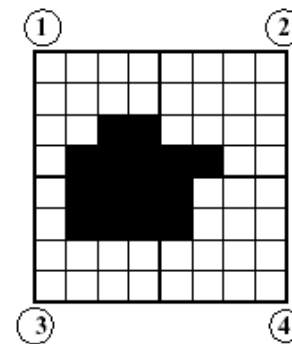
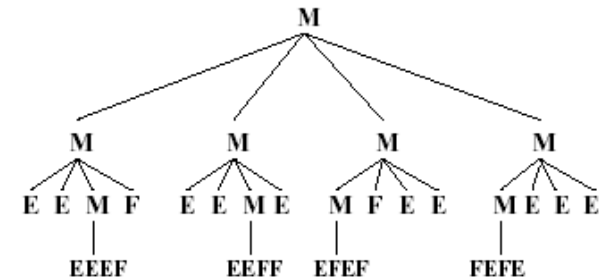


image region



quad tree representation

```

procedure border(S);
{
  for R:= 1 to NLINES
  {
    for C:= 1 to NPIXELS
    {
      LABEL:= S[R,C];
      if new_region(LABEL) then add(CURRENT,LABEL);
      NEIGHB:= neighbors(R,C,LABEL);
      T:= pixeltype(R,C,NEIGHB);
      if T == 'border'
      then for each pixel N in NEIGHB
      {
        CHAINSET:= chainlist(LABEL);
        NEWCHAIN:= true;
        for each chain X in CHAINSET while NEWCHAIN
        {
          if N==rear(X)
          then { add(X,[R,C]); NEWCHAIN:= false }
          if NEWCHAIN
          then make_new_chain(CHAINSET,[R,C],LABEL);
        }
      }
    }
    for each region REG in CURRENT
    {
      if complete(REG)
      then { connect_chains(REG); output(REG); free(REG) }
    }
  }
}

```

	1	2	3	4	5	6	7
1	0	0	0	0	0	0	0
2	0	0	0	0	2	2	0
3	0	1	1	1	2	2	0
4	0	1	1	1	2	2	0
5	0	1	1	1	2	2	0
6	0	0	0	0	2	2	0
7	0	0	0	0	0	0	0

(a) A labeled image with two regions.

Region	Length	List
1	8	(3,2)(3,3)(3,4)(4,4)(5,4)(5,3)(5,2)(4,2)
2	10	(2,5)(2,6)(3,6)(4,6)(5,6)(6,6)(6,5)(5,5) (4,5)(3,5)

(b). The output of the border procedure for the labeled image.

Find the borders of every region of a labeled image S.
S[R, C] is the input labeled image.
NLINES is the number of rows in the image.
NPIXELS is the number of pixels per row.
NEWCHAIN is a flag that is true when a pixel starts a new chain and false when a new pixel is added to an existant chain.

Assume-se que existe uma classe “Background” com regiões disjuntas e cujas fronteiras não precisam de ser identificadas

```

procedure border(S);
{
  for R:= 1 to NLINES
  {
    for C:= 1 to NPIXELS
    {
      LABEL:= S[R,C];
      if new_region(LABEL) then add(CURRENT,LABEL);
      NEIGHB:= neighbors(R,C,LABEL);
      T:= pixeltype(R,C,NEIGHB);
      if T == 'border'
      then for each pixel N in NEIGHB
      {
        CHAINSET:= chainlist(LABEL);
        NEWCHAIN:= true;
        for each chain X in CHAINSET while NEWCHAIN
        {
          if N==rear(X)
          then { add(X,[R,C]); NEWCHAIN:= false }
        }
        if NEWCHAIN
        then make_new_chain(CHAINSET,[R,C],LABEL);
      }
    }
    for each region REG in CURRENT
    {
      if complete(REG)
      then { connect_chains(REG); output(REG); free(REG) }
    }
  }
}

```

Segue da esquerda para a direita, e de cima para baixo

O algoritmo examina 3 linhas por iteração: a linha em processamento, a de cima e a de baixo. Por isso a imagem é estendendida com duas linhas com “Background”, uma em cima e uma em baixo.

O método *neighbors* devolve os vizinhos de do píxel [R,C] com label=LABEL.

O método *pixeltype* considera a vizinhança do píxel [R,C] e valida se é o píxel é “Border”, i.e do contorno de um objeto mas não de “Background”.

Como funciona o método *complete(REG)*?

Detecção e seguimento de contornos – algoritmo de Canny

- Ideias chave:
 - Supressão de não-máximos
 - Seguimento com histerese

Ver também Slides do Cap. 5

Compute thin connected edge segments in the input image.

$I[x,y]$: input intensity image σ : spread used in Gaussian smoothing;
 $E[x,y]$: output binary image

$IS[x,y]$: smoothed intensity image

$Mag[x,y]$: gradient magnitude; $Dir[x,y]$: gradient direction;
 T_{low} is low intensity threshold; T_{high} is high intensity threshold

procedure Canny(I , σ);

```
{
     $IS$  ← image  $I$  smoothed by convolution with Gaussian  $G_\sigma(x,y)$ ;
    use Roberts operator to compute  $Mag[x,y]$  and  $Dir[x,y]$  from  $IS$ ;
    Suppress_Nonmaximal(  $Mag$ ,  $Dir$ ,  $T_{low}$ ,  $T_{high}$  );
    Edge_Detect(  $Mag$ ,  $T_{low}$ ,  $T_{high}$ ,  $E$  );
}
```

procedure Suppress_Nonmaximal(Mag , Dir);

```
{
    define +Del[4] = (1,0), (1,1), (0,1), (-1,1);
    define -Del[4] = (-1,0), (-1,-1), (0,-1), (1,-1);
    for x := 0 to MaxX-1;
    for y := 0 to MaxY-1;
    {
        direction := (  $Dir[x,y]$  +  $\pi/8$  ) modulo  $\pi/4$ ;
        if (  $Mag[x,y] \leq Mag[x,y] + Del(direction)$  ) then  $Mag[x,y] := 0$ ;
        if (  $Mag[x,y] \leq Mag[x,y] + -Del(direction)$  ) then  $Mag[x,y] := 0$ ;
    }
}
```

} procedure Edge_Detect(Mag , T_{low} , T_{high} , E);

```
{
    for x := 0 to MaxX-1;
    for y := 0 to MaxY-1;
    {
        if (  $Mag[x,y] \geq T_{high}$  ) then Follow_Edge(  $x,y,Mag$ ,  $T_{low}, T_{high}, E$  );
    }
}
```

} procedure Follow_Edge(x,y,Mag , T_{low}, T_{high}, E);

```
{
     $E[x,y] := 1$ ;
    while  $Mag[x,y] > T_{low}$  for some Neighbor  $[u,v]$  of  $[x,y]$ 
    {
         $E[u,v] := 1$ ;
         $[x,y] := [u,v]$ ;
    }
}
```

Algoritmo de Canny - Resultados



original



$\sigma = 1$



$\sigma = 4$



original



$\sigma = 1$



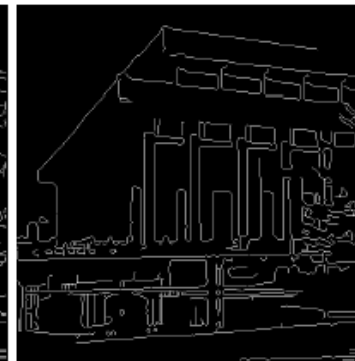
Roberts



original

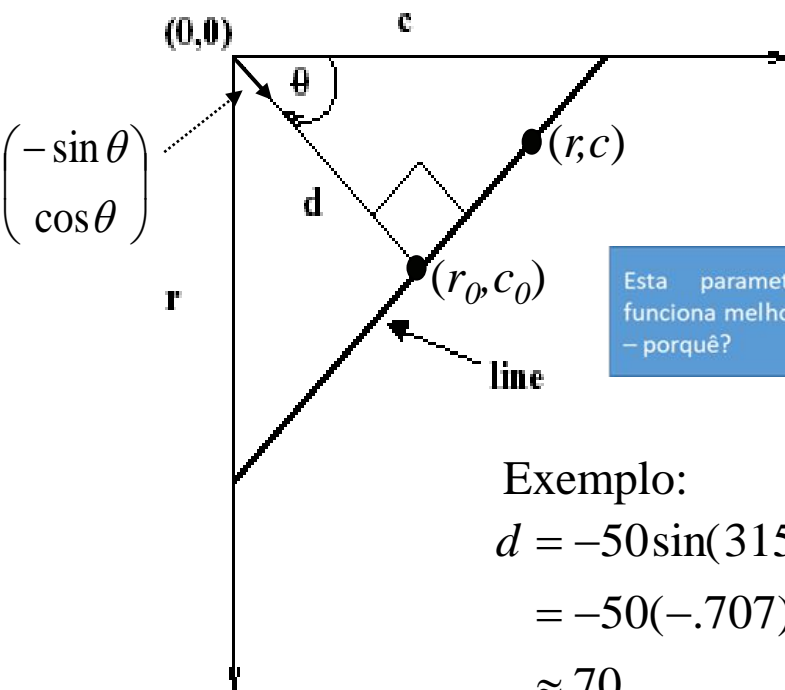


$\sigma = 1$



$\sigma = 2$

Transformada de Hough – Extracção de segmentos de recta



Equação da recta:

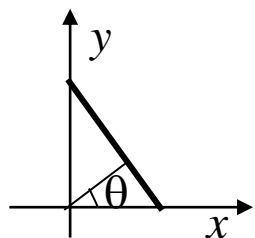
$$d = -r \sin \theta + c \cos \theta$$

Esta parameterização de retas funciona melhor com retas verticais – porquê?

Exemplo:

$$\begin{aligned} d &= -50 \sin(315^\circ) + 50 \cos(315^\circ) \\ &= -50(-.707) + 50(.707) \\ &\approx 70 \end{aligned}$$

Sistema de coordenadas xy:



$$d = x \cos \theta + y \sin \theta$$

$$x \rightarrow c$$

$$y \rightarrow -r$$

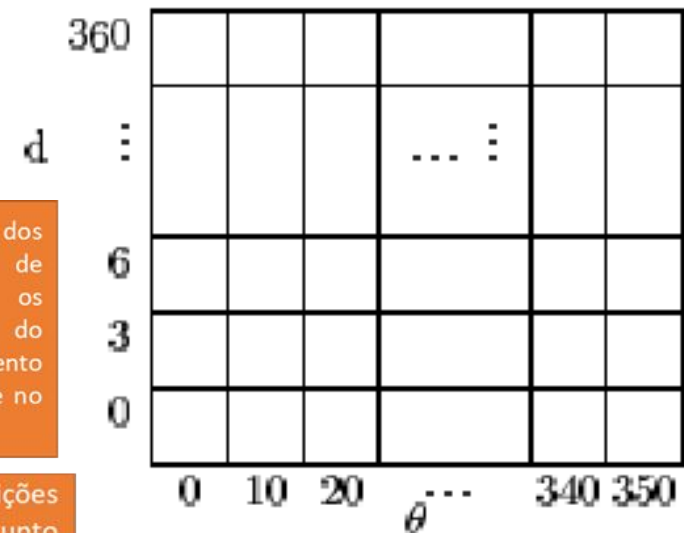
"Histograma" no espaço dos parâmetros: acumula evidência de existência de estruturas com os parâmetros quantizados nos bins do histograma. Pode ter incremento unitário, ou pelo valor do gradiente no ponto de teste.

Manter uma lista com as posições que contribuíram para cada conjunto de parâmetros com evidência não nula.

$$\left\langle \begin{pmatrix} r - r_0 \\ c - c_0 \end{pmatrix}, \begin{pmatrix} -\sin \theta \\ \cos \theta \end{pmatrix} \right\rangle = 0$$

$$\left\langle \begin{pmatrix} r_0 \\ c_0 \end{pmatrix}, \begin{pmatrix} -\sin \theta \\ \cos \theta \end{pmatrix} \right\rangle = d$$

Acumulador $A(d_q, \theta_q)$



H: imagem 256x256

Algoritmo – transformada Hough

Accumulate the straight line segments in gray-tone image S to accumulator A.

S[R,C] is the input gray-tone image.

NLINES is the number of rows in the image.

NPIXELS is the number of pixels per row.

A[DQ,THETAQ] is the accumulator array.

DQ is the quantized distance from a line to the origin.

THETAQ is the quantized angle of the normal to the line.

```
procedure accumulate_lines(S,A);
{
  A := 0;
  PTLIST := NIL;
  for R := 1 to NLINES
    for C := 1 to NPIXELS
      {
        DR := row_gradient(S,R,C);
        DC := col_gradient(S,R,C);
        GMAG := gradient(DR,DC);
        if GMAG > gradient_threshold
          {
            THETA := atan2(DR,DC);
            THETAQ := quantize_angle(THETA);
            D := abs(C*cos(THETAQ) - R*sin(THETAQ));
            DQ := quantize_distance(D);
            A[DQ,THETAQ] := A[DQ,THETAQ]+GMAG;
            PTLIST(DQ,THETAQ) := append(PTLIST(DQ,THETAQ),[R,C])
          }
      }
}
```

Find the point lists corresponding to separate line segments.
A[**DQ**,**THETAQ**] is the accumulator array from `accumulate_lines`.
DQ is the quantized distance from a line to the origin.
THETAQ is the quantized angle of the normal to the line.

```

procedure find_lines;
{
V := pick_greatest_bin(A,DQ,THETAQ);
while V > value_threshold
{
list_of_points := reorder(PTLIST(DQ,THETAQ));
for each point [R,C] in list_of_points
    for each neighbor (R',C') of [R,C] not in list_of_points
    {
        DPRIME := D(R',C');
        THETAPRIME := THETA(R',C');
        GRADPRIME := GRADIENT(R',C');
        if GRADPRIME > gradient_threshold
            and abs(THETAPRIME - THETA) ≤ 10
        then {
            merge(PTLIST(DQ,THETAQ),PTLIST(DPRIME,
                THETAPRIME));
            set_to_zero(A,DPRIME,THETAPRIME);
        }
    }
}
final_list_of_points := PTLIST(DQ,THETAQ);
create_segments(final_list_of_points);
set_to_zero(A,DQ,THETAQ);
V := pick_greatest_bin(A,DQ,THETAQ);
}
}

```

Algorithm 27: O’Gorman/Clowes Method for Extracting Straight Lines

Exemplo

Pré-processamento

	1	2	3	4	5
1	0	0	0	100	100
2	0	0	0	100	100
3	0	0	0	100	100
4	100	100	100	100	100
5	100	100	100	100	100

gray level image

	1	2	3	4	5
1	0	0	300	300	0
2	0	0	300	300	0
3	0	0	200	200	0
4	0	0	0	100	0
5	0	0	0	0	0

column gradient

	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	300	300	200	100	0
4	300	300	200	100	0
5	0	0	0	0	0

row gradient

	1	2	3	4	5
1	-	-	0	0	-
2	-	-	0	0	-
3	90	90	45	26.6	-
4	90	90	90	45	-
5	-	-	-	-	-

THETA

	1	2	3	4	5
1	-	-	0	0	-
2	-	-	0	0	-
3	90	90	40	20	-
4	90	90	90	40	-
5	-	-	-	-	-

THETAQ

	1	2	3	4	5
1			3	4	
2			3	4	
3	3	3	4.2	4.8	
4	4	4	4	5.6	
5					

D

	1	2	3	4	5
1			3	3	
2			3	3	
3	3	3	3	3	
4	3	3	3	3	
5					

DQ

Acumulador

DQ	360						
:							
6							
3	4		1		2		5
0							
	0	10	20	30	40	...	90

accumulator A

DQ	360						
:							
6							
3	♣		◇		♠		♥
0							
	0	10	20	30	40	...	90

PTLIST

- ♣ (1,3)(1,4)(2,3)(2,4)
- ◇ (3,4)
- ♠ (3,3)(4,4)
- ♥ (3,1)(3,2)(4,1)(4,2)(4,3)

Accumulate the circles in gray-tone image S to accumulator A .

$S[R, C]$ is the input gray-tone image.

$NLINES$ is the number of rows in the image.

$NPIXELS$ is the number of pixels per row.

$A[R, C, RAD]$ is the accumulator array.

R is the row index of the circle center.

C is the column index of the circle center.

RAD is the radius of the circle.

```
procedure accumulate_circles(S,A);
{
  A := 0;
  PTLIST := 0;
  for R := 1 to NLINES
    for C := 1 to NPIXELS
      for each possible value RAD of radius
        {
          THETA := compute_theta(S,R,C,RAD);
          R0 := R - RAD*cos(THETA);
          C0 := C + RAD*sin(THETA);
          A[R0,C0,RAD] := A[R0,C0,RAD]+1;
          PTLIST(R0,C0,RAD) := append(PTLIST(R0,C0,RAD),[R,C])
        }
    }
}
```

Algorithm 28: Hough Transform for Finding Circles

Transformada de Hough Generalizada

- A TH pode ser estendida para qualquer curva com expressão analítica da forma $f(\mathbf{x}, \mathbf{a}) = 0$,
 - \mathbf{x} é um vector 2D com as coordenadas de um ponto da curva
 - \mathbf{a} é um vector de parâmetros
- Algoritmo genérico:
 - Inicializar acumulador $A(\mathbf{a})$ a zero
 - Por cada pixel \mathbf{x} na imagem, determinar valores de \mathbf{a} , tais que $f(\mathbf{x}, \mathbf{a}) = 0$; incrementar acumulador: $A(\mathbf{a}) = A(\mathbf{a}) + 1$
 - Máximos locais de A correspondem a curvas f na imagem